SR5

Netflex
Software Architecture Document

Version 1.1

Revision History

Date	Version	Description	Author
25/Nov/23	1.0	Finished the following sections: 1,2,3	Đinh Quang Phong
28/Nov/23	1.1	Finished the remaining required sections for PA3 and made some adjustments to the document as a whole	Phạm Quốc Duy, Đinh Quang Phong, Nguyễn Tấn Lộc, Tăng Tường Thoại, Trần Thành Duy

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This document provides a comprehensive sight of the Netflex movie streaming website system as well as its own external and integral functionalities. Addtionally, several architectural views were used specifically to describe different aspects of the structure. Nonetheless, a list of goals and final requirements are depicted detailedly.

1.3 Definitions, Acronyms and Abbreviations

No.	Definitions/Acronyms/Abbreviations	Explanation
1	MVC	Model-View-Controller
2	DDOS	Distributed Denial-Of-Service
3	API	Application Program Interface

1.4 References

- Software Architecture Document template
- Netflex's Vision Document
- Netflex's Use Case Specification Document

1.5 Overview

- Introduction: the document provides a brief entry to Netflex architect document
- Architectural Goals and Constraints: list out the high-level goals and requirements that affect the structure of Netflex website
- Use-case Model: states the use case and scenarios from different stakeholder's views
- Logical View: presents the detailed arrangement of Netflex system and illustrating the connection of the components and their shared responsibility for achieving goals
- Deployment: showcases the actual implementation of software components onto physical hardware
- Implementation View: the final product that has been implemented of Netflix streaming website

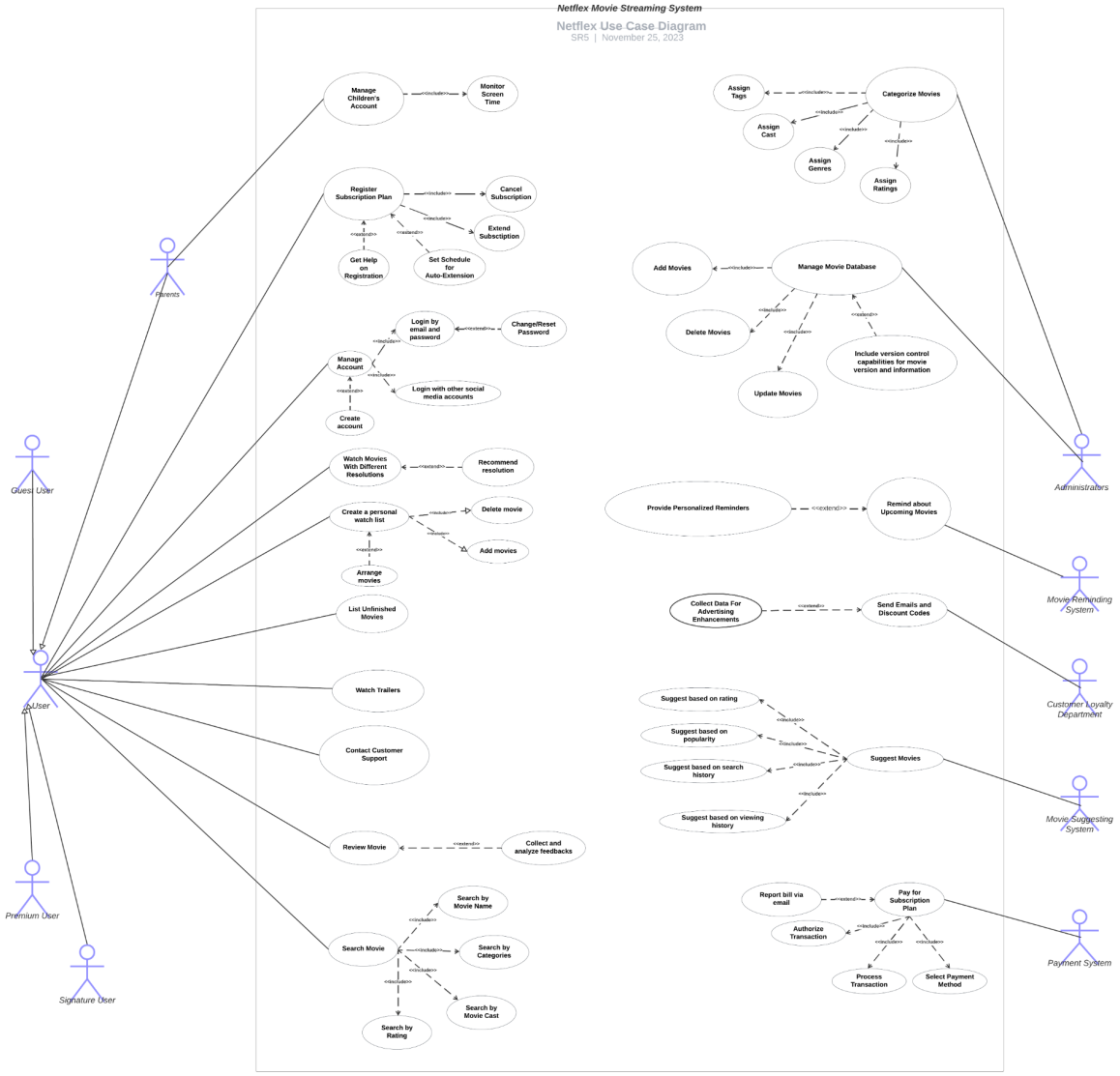
2. Architectural Goals and Constraints

- Programming language: The website must be built by using the following coding languages: HTML, CSS, Javascript
- For framework:
  - Front end: Bootstrap 5

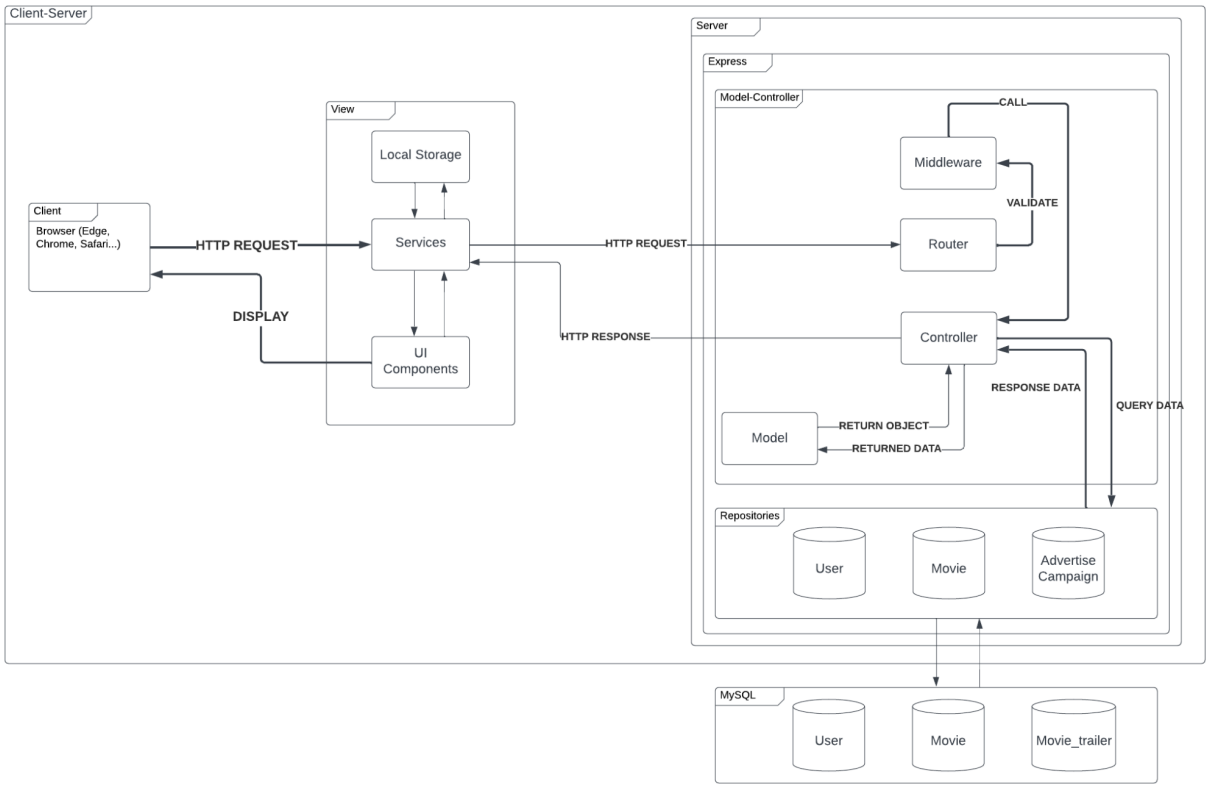
- Back end: Express - NodeJS

- For database: MySQL, Clever cloud as database hosting service
- Platform: The website front page is required to be flexible so that it could fit every device's screen which has an internet connection to the site.
- Performance:
  - Movie or trailer must be shown from 60 to 100 frames per second, there also shouldn't be lag or stuttering exist on vast amount of users over the whole server
  - After a client click on playing video button the server should start streaming within 0.1 second
  - The website should be able to handle a large number of concurrent users and each page has to load within 2 seconds, especially during peak times and new movie releases.
  - On searching for movies from the database, the action is forced to be no more than 1 second
  - In parental mode, the system have to cease the children on watching over 180 minutes as default if parents haven't set the time limit
  - As many movies are categorized or filtered in different genres, the action of the system is asked to be no more than 0.2 seconds.
  - Server crashing should be resolved from 3 to 5 minutes
  - Even though maintenance is an indispensable process in such movie websites like Netflex, there must be an announcement for all users about the upcoming update and the event should last under one hour.
- Security:
  - Netflex website must be capable of dealing with DDOS attacks.
  - During payment or any kind of transactions, if the server suddenly collapses or client loses connection the status of the database and the account balance must remain unchanged, all of which are required to be conducted under 3 seconds.

3. Use-Case Model



4. Logical View

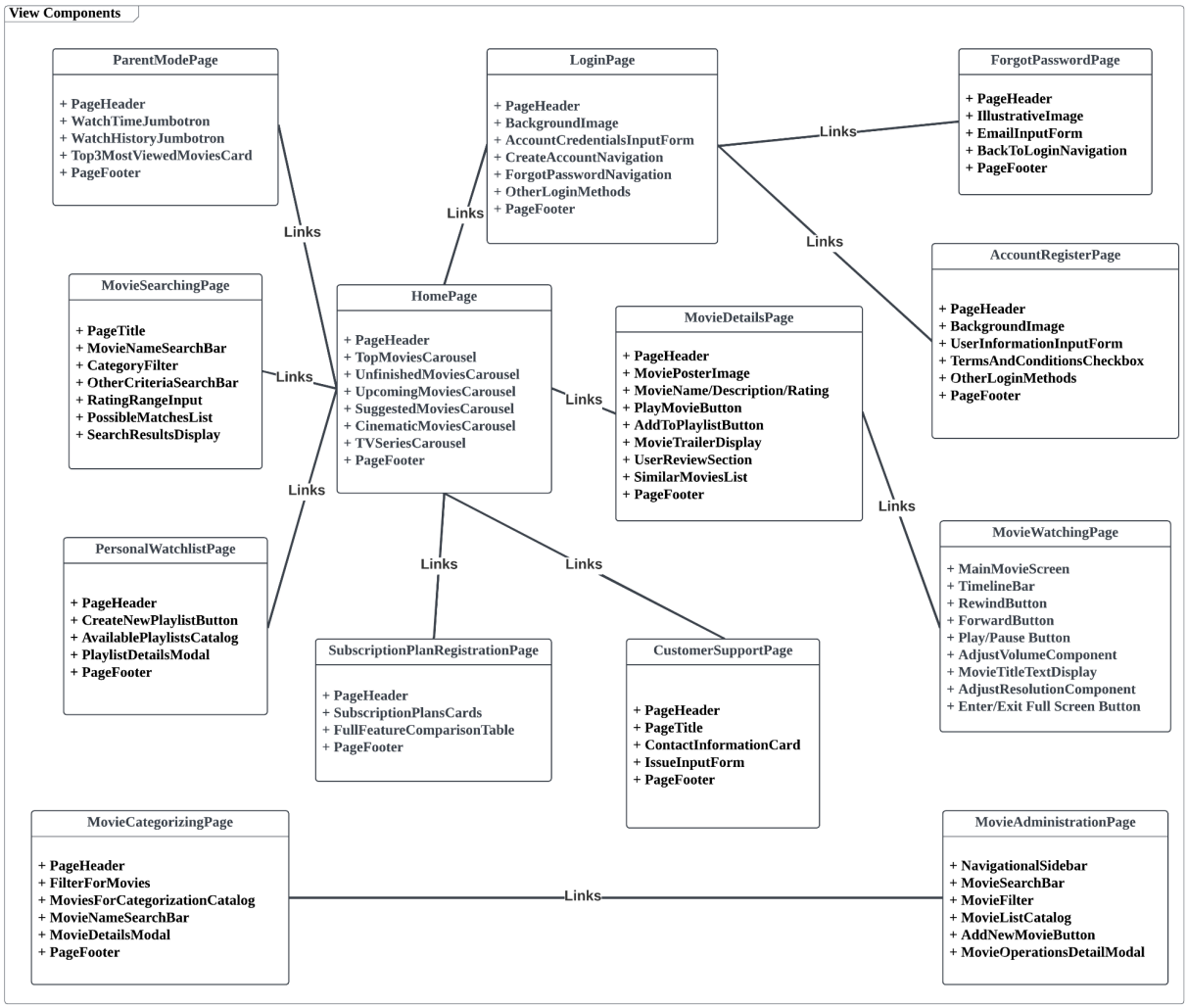


From a general perspective, MVC (Model-View-Controller) architecture is applied for the overall implementation of the Netflex movie streaming system. Extracting from the Logical View package diagram, it can be seen that the architecture consists of the following components:

- **Client Side:**
  - **View Package:**
    - Responsible for managing the user interface and its related components and screens.
    - Communicates with the Controller package to display relevant data and respond to user inputs.
- **Server Side:** The Server component forms the backend of Netflex, handling authentication, database interaction, and processing user requests from the client.
  - **Controller Package:**
    - Establishes communication between View and Model packages.
    - Manages user interactions, processes inputs and carries out appropriate actions based on requests.
    - Receives data from the Model package and updates the View, ensuring a dynamic display of information.
  - **Model Package:**
    - Houses data entities and the business logic of the application including movies, user preferences, reviews, etc.
    - Retrieves, manipulates and validates data..
  - **Middleware Package:**
    - Manages authentication, validation tasks and acts as an intermediary for different processes

- **Repository Package:**
  - Establishes and maintains a robust Database Management System (DBMS) to store diverse data like movie details, user profiles, movie trailers…etc.
  - Enabling data transition to Controllers

4.1 Component: Views (UI Component)



Key Classes Explanation:

HomePage: Is the most crucial part in the Views component, containing the following sub-components:

- *PageHeader*: a simple header responsible for displaying the navigation, account tab and search icon
- *TopMoviesCarousel, UnfinishedMoviesCarousel, UpcomingMoviesCarousel, SuggestedMoviesCarousel, CinematicMoviesCarousel, TVSeriesCarousel*: simple carousels components responsible for displaying lists movies including top movies, unfinished movies, upcoming movies, suggested movies, cinematic movies and TV Series respectively

- *PageFooter:* a simple footer showing a brief description of the application, navigation bar as well as social media links of Netflex

ParentModePage: Is linked to HomePage and contains the following sub-components:

- *WatchTimeJumbotron*: a simple jumbotron displaying the average watch time of the children's account
- *WatchHistoryJumbotron*: a simple jumbotron displaying most recent watch history of the children's account
- *Top3MostViewedMoviesCard*: a simple collection of cards containing information of movies that the children account spent the most time watching

MovieSearchingPage: Plays an important role in the Views component, containing the following sub-components:

- *PageTitle*: a simple title displayed to the user
- *MovieNameSearchBar*: allows the user to search for specific movie(s) by name
- *CategoryFilter*: allows the user to filter movies by categories
- *OtherCriteriaSearchBar*: allows the user to search by other criterias (cast, directors…)
- *RatingRangeInput*: displays a range of ratings (one to five stars) for the user to find movies with such rating
- *PossibleMatchesList*: list of movies that may match the search criteria(s) (auto-filling)
- *SearchResultsDisplay*: list of movies that matched the search criteria(s)

PersonalWatchlistPage: Is connected to HomePage and contains the following sub-components:

- *CreateNewPlaylistButton*: enables the user to create a new watchlist
- *AvailableWatchlistsCatalog*: displays a series of personal watchlists belonging to the user
- *PlaylistDetailsModal*: displays the details of a certain watchlist including list of movies and actions available to that list (add movies, delete movies, delete list…)

SubscriptionPlanRegistrationPage: Is an essential part in the Views component, linked to HomePage and contains the given sub-components:

- *SubscriptionPlanCards*: displays the subscription packs in the form of cards containing information such as: plan name, price, description, purchase button, …etc
- *FullFeatureComparisonTable*: displays the detailed feature comparison between various subscription plans in the form of a table

CustomerSupportPage: Is linked to HomePage and contains the following sub-components:

- *ContactInformationCard*: a simple card displaying basic contact information of customer support team such as email and hotline number
- *IssueInputForm*: displays a form for the user to enter information about the issue they encountered

MovieDetailsPage: Plays a crucial part in the Views component, linked to HomePage and contains the given sub-components:

- *MoviePosterImage*: displays a poster of the movie
- *MovieName/Description/Rating*: a display combination of the movie's name, description and ratings
- *PlayButton, AddToPlayListButton*: allows the user to play and add the movie to their playlist, respectively
- *MovieTrailerDisplay*: displays the trailer of the movie
- *UserReviewSection*: a section devoted to displaying users' reviews on the movie
- *SimilarMoviesList*: displays a list of movies with similar characteristics to this movie

MovieWatchingPage: Is linked to MovieDetailsPage, containing the following sub-components:

- *MainMovieScreen*: main display where the movie is shown to the user
- *TimelineBar*: a simple bar allowing the user to move to specific parts of the movie
- *RewindButton, ForwardButton*: simple buttons allowing the user to rewind, fast-forward respectively
- *Play/Pause Button*: a simple button that lets the user pause/continue the movie while watching
- *AdjustVolumeComponent*: allows the user to adjust the volume of the movie
- *MovieTitleTextDisplay*: simple display of the movie title
- *AdjustResolutionComponent*: allows the user to adjust the movie's resolution with a limit
- *Enter/Exit Fullscreen Button*: allows the user to enter/exit fullscreen mode

LoginPage: Is an important part of the Views component, linked to HomePage and contains the following sub-components:

- *BackgroundImage*: an image representing a background image for Netflex
- *AccountCredentialsInputForm*: allows the user to enter their email, password to login
- *CreateAccountNavigation, ForgotPasswordNavigation*: allows the user to navigate to the create account and password recovery page accordingly
- *OtherLoginMethods*: display other login methods to the user (social media platforms)

ForgotPasswordPage: Is linked to LoginPage and contains the given sub-components:

- *EmailInputForm*: a simple form allowing the user to enter their email for password recovery
- *BackToLoginNavigation*: allows the user to navigate back to the login page

AccountRegisterPage: Is linked to LoginPage and contains the following sub-components:

- *UserInformationInputForm*: a simple form displayed to retrieve the user's account's basic information
- *TermsAndConditionsCheckbox*: ensuring that the user agrees to the terms and conditions before continuing

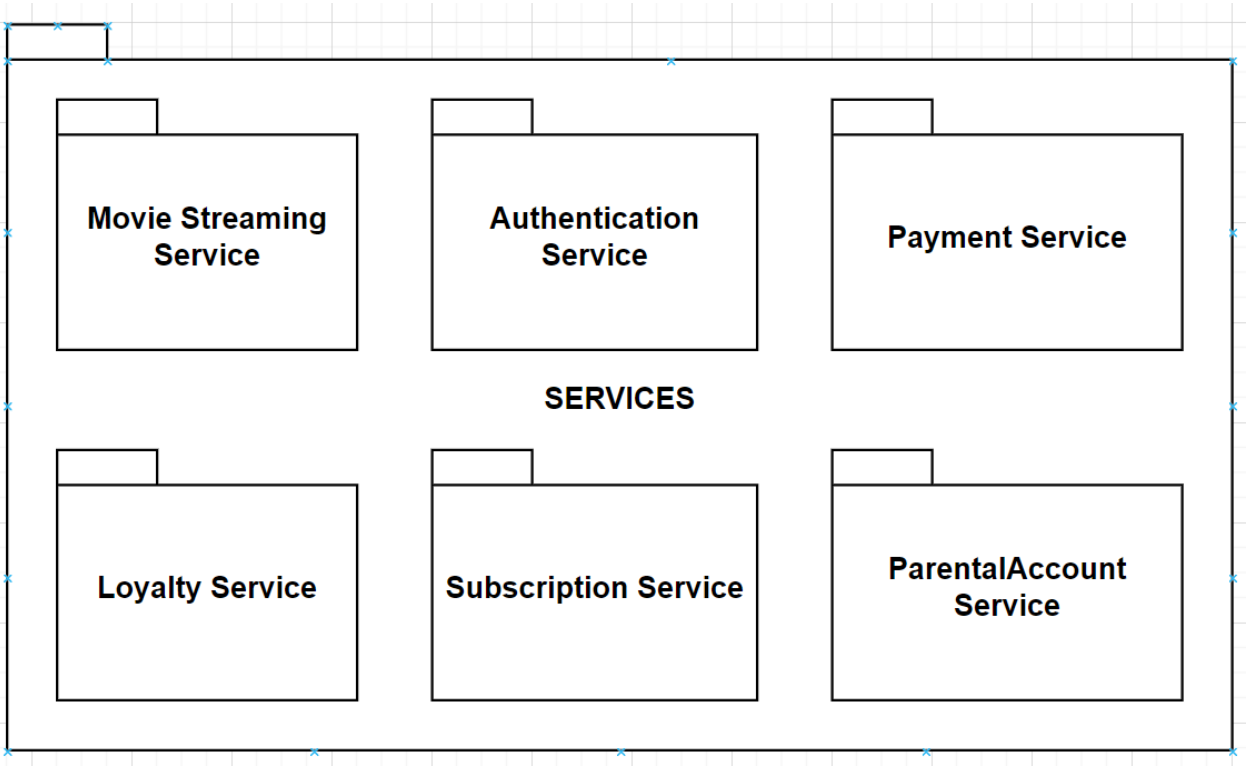
MovieCategorizingPage: Plays a crucial role in the Views component serving the administrator (hereby called admin) and has some sub-components:

- *FilterForMovies*: allows the admin to filter for specific movies before categorizing them
- *MoviesForCategorizationCatalog*: a main display of movies that require categorizing in a list
- *MovieNameSearchBar*: allows the admin to search for a specific movie
- *MovieDetailsModal*: enabling the admin to look into the details of a movie and perform specific tasks: add tag, add category, …etc

MovieAdministrationPage: Plays a crucial role in the Views component serving the admin, is linked to MovieCategorizing Page and contains the following sub-components:

- *NavigationalSidebar*: a simple sidebar providing navigation to different sections of the admin dashboard
- *MovieSearchBar*: allows the admin to search for a specific movie
- *MovieFilter*: allows the admin to filter movies by various criterias
- *MovieListCatalog*: a main display of the movies in the system in the form of a list
- *AddNewMovieButton*: a simple button letting the admin add a new movie to the system
- *MovieOperationDetailModel*: displaying information about a specific movie and enabling specific CRUD operations on the movie

4.2 Component: Services



Key Class Diagram

Key Classes Explanations

4.2.1. Movie Streaming Service

- The **Movie Streaming Service** is responsible for handling streaming movies to users, and is capable of streaming with various resolutions, playback speed, etc.
- It brings along with filter functionality, searching movies by name and also reviews.

4.2.2. Authentication Service

- The **Authentication Service** is responsible for dealing with authenticated users who are participating in the system. It is also responsible for handling many operations which are related to the user's account, for example, changing password, registering a new account, forgetting password,etc.

4.2.3. Payment Service

- The **Payment Service** is responsible for the user's purchasing subscription.
- It can be appealing in various forms. For instance, using a credit card to pay, using an e-wallet to pay, …

4.2.4. Loyalty Service

- The **Loyalty Service** is responsible for managing the user's interaction with the app or system.
- It tracks user’s experiences while using the app, what users like or interested in, …and then perform marketing actions.
- Additionally, Loyalty service is also responsible for taking user requests or complaints and helping them to resolve their problems.

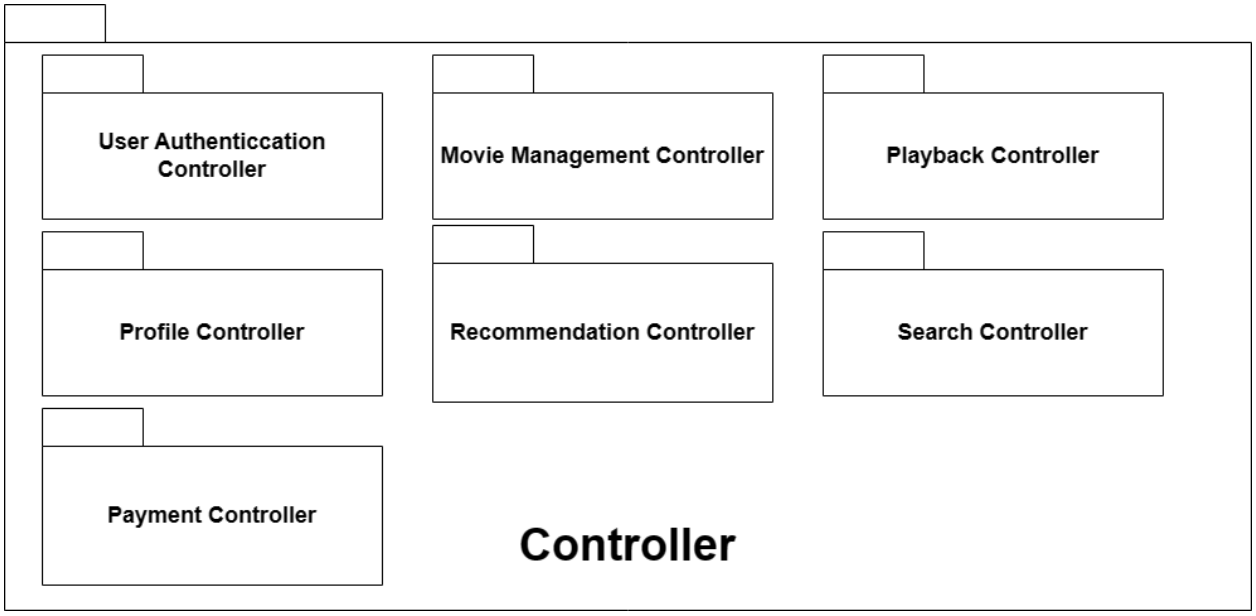
4.2.5. Subscription Service

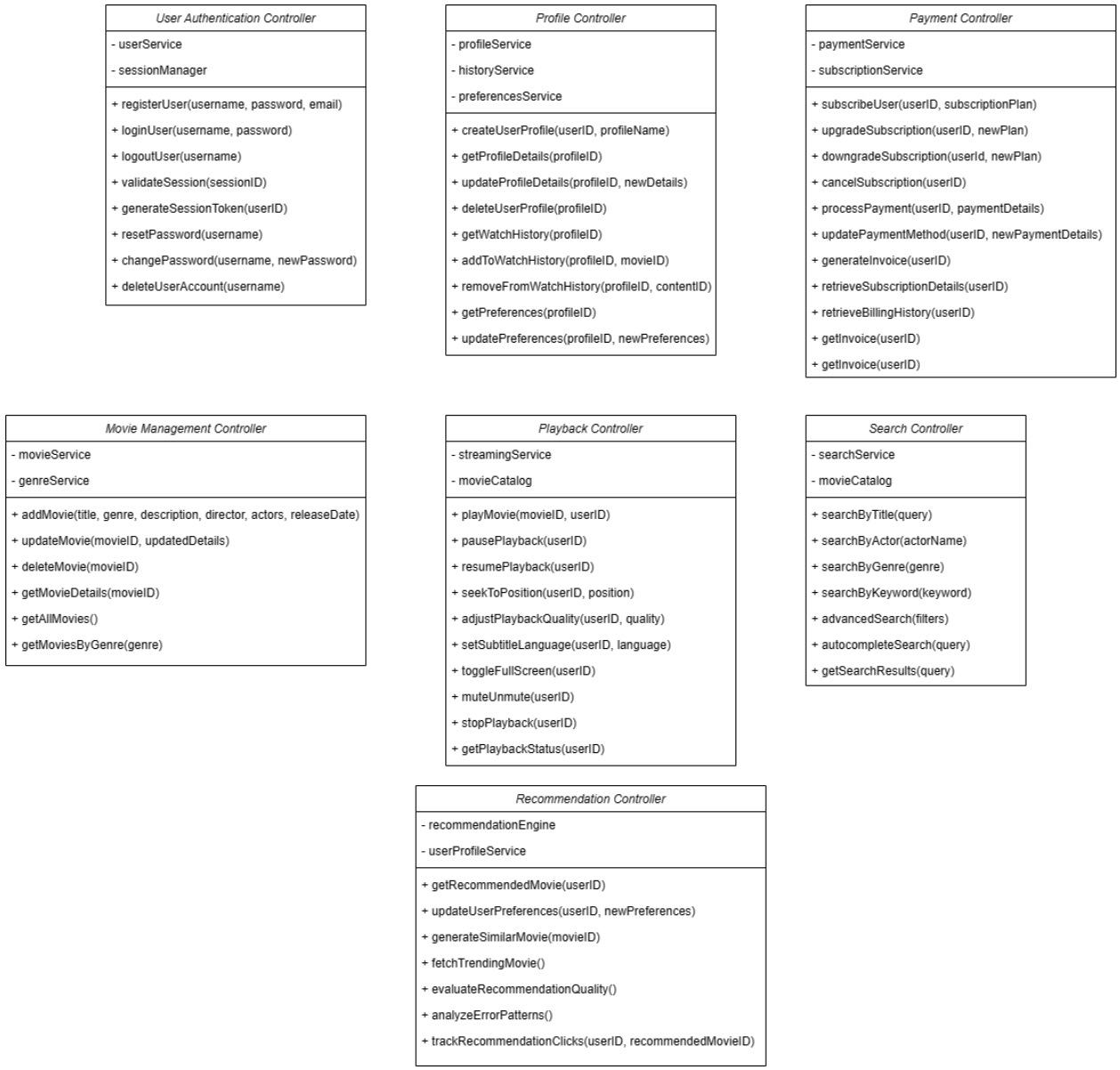
- The **Subscription Service** is responsible for managing the user’s subscription package.
- It allows the users to register, cancel, update their subscription package which is dependent on their desire.

4.2.6. ParentalAccount Service

- The **Parental Account Service** is responsible for managing the users account usage. For instance, limiting usetime, content limited by ages, etc.
- The users need to manually controller what account they want to controller after they have proven their authorities.

4.3 Component: Controllers





Key Classes Explanations

User Authentication Controller:

- The **User Authentication Controller** in Netflex is an important part located in the Controller section.
- It's in charge of key user actions such as logging in, signing up, and logging out.
- This controller handles requests to confirm users' identities and connects what users see on the screen to the system that checks if they're genuine.
- It works closely with the user interface (UI), managing the information users input and ensuring smooth data flow between the system behind the scenes (Model) and what's displayed on the Netflex screen.

Profile Controller:

- The **Profile Controller** handles user profiles.
- It takes care of creating, modifying, and deleting user profiles.
- This controller manages user preferences, watch history, and favorite lists.
- It works closely with what users see on the screen (UI) and helps process the information users input.

Payment Controller:

- The **Payment Controller**, manages user subscriptions, billing cycles, and payment processes.
- It's responsible for handling subscription upgrades, downgrades, and updates to payment methods.
- This controller facilitates smooth interactions between users and the payment system, ensuring secure and efficient subscription management.
- It works in harmony with the user interface (UI), processing user input related to payments.

Movie Management Controller:

- The **Movie Management Controller** oversees the administration of movies within the platform.
- It handles functions such as adding, modifying, and removing movies from the database.
- This controller manages metadata related to movies, including titles, genres, descriptions, and other essential details.
- It collaborates with the user interface (UI), processing input related to movie management.

Playback Controller:

- The **Playback Controller** controls the streaming and playback features.
- It manages the video player settings, buffering, quality adjustments, and playback controls.
- This controller enables users to play, pause, resume, and stop movies, as well as seek specific positions within the video.
- It collaborates with the user interface (UI) to process user interactions related to video playback.

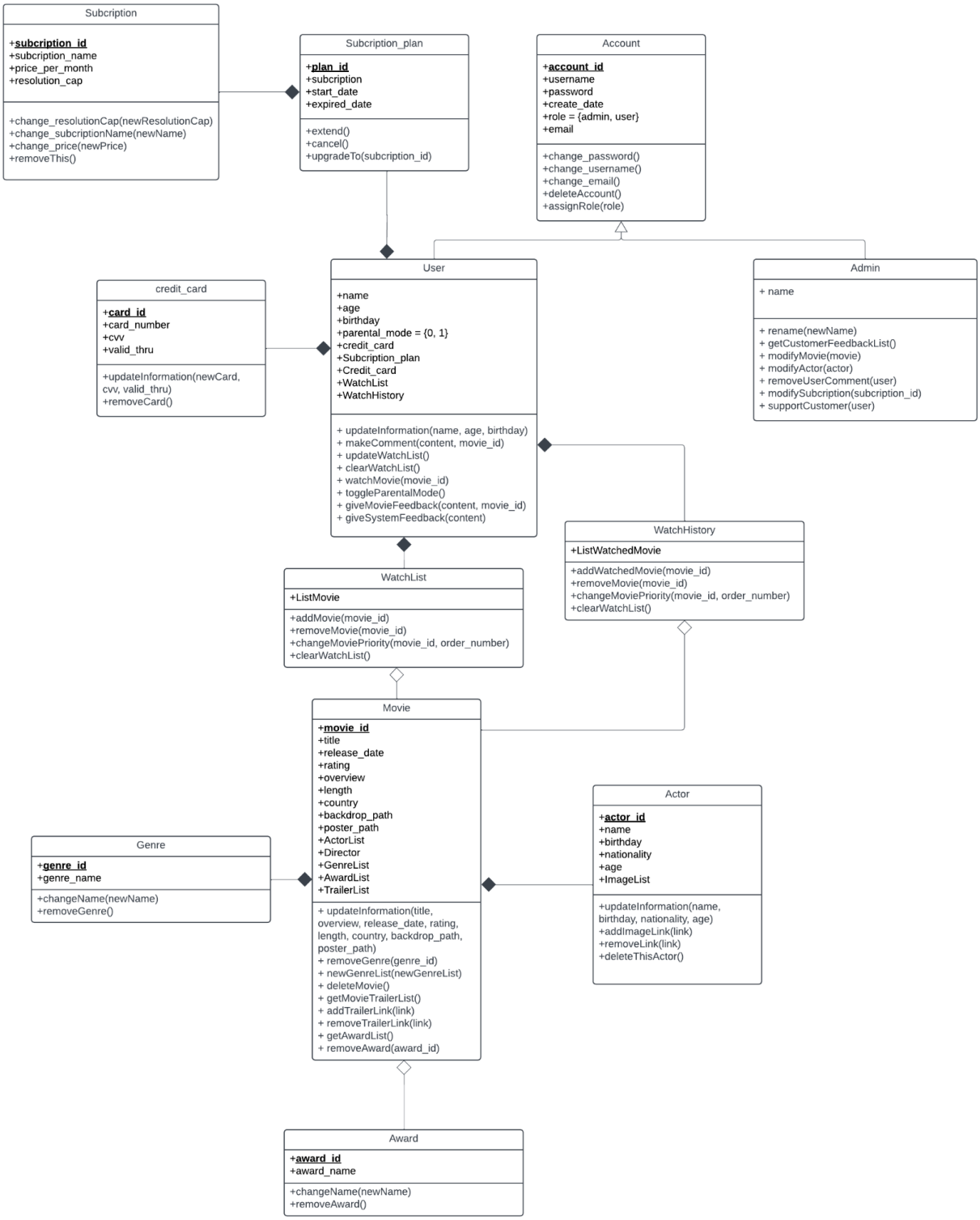
Search Controller

- The **Search Controller** is responsible for handling various search functionalities.
- It manages searches for movies, TV shows, actors, genres, and other movies.
- This controller interacts with the search service to retrieve relevant movies based on user queries.
- It collaborates closely with the user interface (UI), processing user inputs for searches.

Recommendation Controller

- The **Recommendation Controller**, governs the logic for generating personalized movie recommendations.
- It communicates with the recommendation engine based on user preferences and history.
- This controller plays a crucial role in offering tailored movie suggestions to users.
- It collaborates closely with the user interface (UI), processing user preferences and interactions.

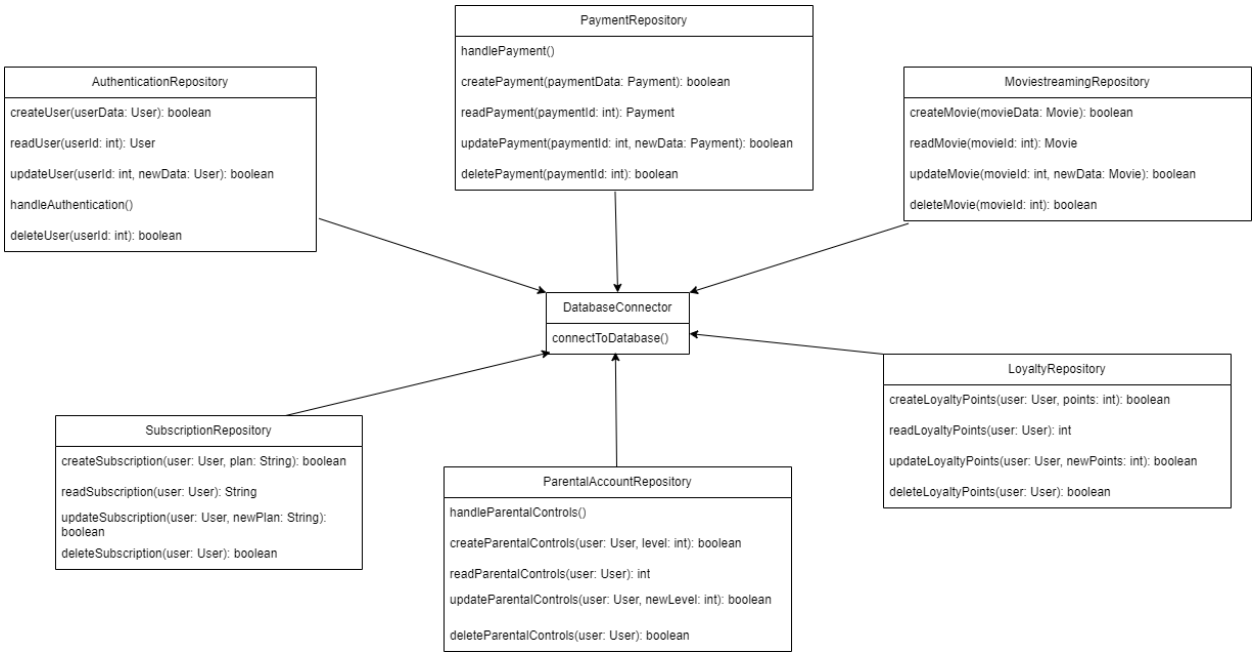
4.4 Component: Models



Some key classes:

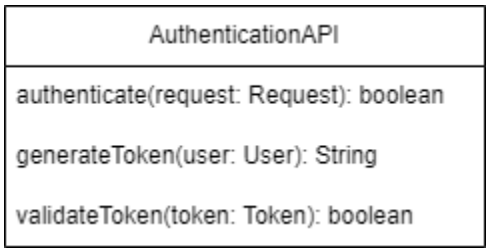
- **Account:**
  - The abstract class of **User** and **Admin**
  - Responsible for holding the vital authentication data for User and Admin
  - Showcases the limited use of User and Admin by assigning roles
  - Some vital data for instance: username, password, email can only be modified by the user or admin of such account
- **User:**
  - Responsible for managing user's personal information.
  - It's the popular role for most stakeholders such as: Student, Practitioner teacher, Internet users, …
  - Conducting user operations by interacting with other components
  - It collaborates with **Movie** class to give users many key features, specifically watch Movie, add to Watchlist, …
  - **User** can toggle the parental mode which will limit the watching time as well as movie genres.
- **Admin:**
  - It's the key role that manipulates the whole Netflex site
  - Admin can interfere in some use cases provided by the structure for example: removeUserComment, modify Movie, Subscription, Actor …
- **credit_card:**
  - Each **User** may have one credit card in order to upgrade, extend or cancel their subscription plan
  - It helps the transaction validation much easier for users by recycling the data for future uses.
- **subscription:**
  - This is the solid data of Netflex including 3 types of subscriptions are Free, Premium, Signature respectively.
  - Only **Admin** can modify this class and save it to database
- **Movie:**
  - This is the model for movie data in the database as well as the real time data for a movie while running on the server.
  - **Movie** is the second most essential role in Netflex streaming sites, from which the background information can be fetched and uploaded namely **Actors, Director, Trailer, title, overview, …**
  - **Admin** is the maintainer for this class and its data within the database
- **Actor:**
  - This class represents the actor, actress and director for each movie.
  - Each **Actor** has personal information around it but only for advertisements such as *name, birthday, nationality, age, Image list…*

4.5 Component: Repository



- **DatabaseConnector:**
  - Ensures connection and interaction with the database to store and retrieve information regarding video content, users, and analytics.
- **MovieStreamingRepository:**
  - Function: Manages CRUD operations related to movie information.
  - Specific Tasks: Creates, reads, updates, and deletes information about movies in the system.
- **ParentalAccountRepository:**
  - Function: Manages parental control settings and related information.
  - Specific Tasks: Creates, reads, updates, and deletes control information, manages restrictions or rules for child accounts as requested by parents.
- **SubscriptionRepository:**
  - Function: Manages information about subscription registration and cancellation for service packages.
  - Specific Tasks: Creates, reads, updates, and deletes subscription information for users.
- **LoyaltyRepository:**
  - Function: Manages the loyalty points feature and benefits for users.
  - Specific Tasks: Creates, reads, updates, and deletes accumulated point information, manages user benefits or promotional programs.
- **PaymentRepository:**
  - Function: Manages payment operations within the system.
  - Specific Tasks: Creates, reads, updates, and deletes transactional payment information, including payment processing, refunds, etc.
- **AuthenticationRepository:**
  - Function: Manages user authentication and user-related information.
  - Specific Tasks: Creates, reads, updates, and deletes user information, including authentication, account management, personal information, etc.

4.6 Component: Middleware



- **AuthenticationAPI**:
  - Responsible for checking whether a user is a human or bot, and preventing spam that could overload the website.

5. Deployment

6. Implementation View